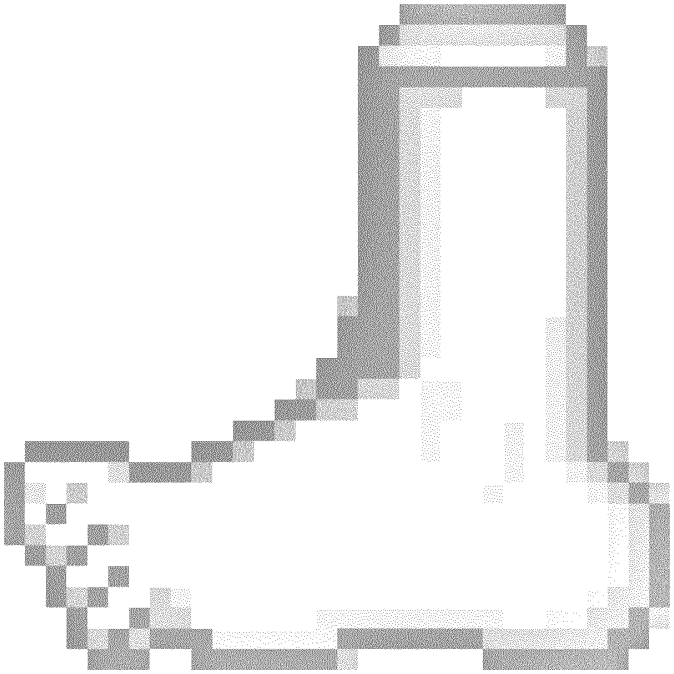


Python, la non-documentation

Mathieu Lecarme

Sommaire

Préambule

> La vie est courte
> Les livres ont trop de papiers
> Code ici et maintenant !
>
> — Lao Tseu, Citation d'une discussion sur IRC

Ce fascicule devrait permettre aux personnes trop pressées, trop enthousiastes, de faire leurs premiers pas de développeurs Python, tout de suite, les mains dans le cambouis.

Les pas suivants se feront en lisant de la vraie documentation, personne ne peut y couper.

Dans chaque chapitre, la section ハッキング **Hakkingu** donne des astuces pour faire le malin et attiser la curiosité.

Supports

Cette non-documentation est disponible sur différentes supports :

- Web [https://nondocumentation.garambrogne.net/]
- PDF [https://nondocumentation.garambrogne.net/python_non_documentation.pdf]

Licence

Les sources de ce document athoune/python_non_documentation [https://github.com/athoune/python_non_documentation] sont sous licence Creative Common.

Licence

Python

Python est un **langage de programmation**.

Il fait maintenant partie de l'enseignement des mathématiques en France.

Il est utilisé dans le calcul scientifique, l'intelligence artificielle, le web, et beaucoup d'autres choses.

Interpréteur interactif

L'interpréteur interactif permet de taper des commandes Python, une à une et d'afficher le résultat (et les erreurs quand on se trompe). Il est possible de remonter dans l'historique, pour reprendre une ligne précédante, sans avoir à tout retaper.

L'interpréteur interactif de Python est disponible en ligne (mais pas sur les navigateurs sans clavier) sans avoir besoin d'installer quoi que ce soit : Pyodide interactif [https://pyodide.org/en/stable/console.html].

Interpréteur

L'interpréteur Python va lire des fichiers contenant du code Python.

Ces fichiers seront édités avec un éditeur de texte.

Des éditeurs en ligne, combinant un éditeur de texte et un interpréteur, permettent d'éditer du code, confortablement, sans rien devoir installer. MyCompiler [https://www.mycompiler.io/fr/new/python] est l'un d'eux.

> ハッキング *Hakkingu*
>
> Python est un langage **interprété**, **faiblement typé**.
>
> Python a un tutorial officiel [https://docs.python.org/fr/3.13/tutorial/interpreter.html].
>
> La calcualtrice Numworks fournit un simulateur en ligne [https://www.numworks.com/fr/simulateur/]. L'interpréteur intéractif est nommé "Console d'exécution".
>
> 
>
> La plupart des interpréteurs intéractifs fournissent de la complétion. La touche →| propose de compléter le début de mot que l'on a tapé.
>
> Il existe des interpréteurs interactifs plus pratiques que celui par défaut, comme iPython [https://ipython.org/].
>
> Pyodide, qui permet de lancer du Python sur Internet, tourne dans le navigateur web, avec **web assembly**, sans solliciter le serveur web.
>
> Le nom Python ne vient pas du serpent, mais des Monty Python [https://fr.wikipedia.org/wiki/Monty_Python], une célèbre troupe d'humoristes anglais.

Types

Python utilise des valeurs de différents types.

Nombres

int : Entier

```
>>> 1+2
3
>>> 1-2
-1
```

float : Nombre réel

```
>>> 1/2
0.5
```

Python utilise la notation anglaise, avec un point comme séparateur et non une virgule.

Python transforme un `int` en `float` lorsque c'est nécessaire.

Opérateurs

Python fournit tous les opérateurs de base et gère les priorités. Il est possible de mettre plusieurs espaces avant/après les opérateurs.

```
>>> 1+2*3
7
>>> (1+2)*3
9
>>> 3*7
21
>>> # ** : puissance
>>> 3**2
9
```

bool : Booléen

`True` pour vrai, `False` pour faux.

Les `bool` ont leurs propres opérateurs :

```
>>> True or False
True
>>> True and False
False
>>> not False
True
```

Les comparateurs permettent de créer un booléen comparant deux valeurs : une **clause**.

```
>>> 1 == 2
False
>>> 1 != 2
True
>>> 1 > 2
False
>>> 1 <= 2
True
```

Pour distinguer le = qui sert à assigner une valeur, une comparaison d'égalité va utiliser le **double égal** ==. Par convention, l'opérateur pour "différent" est **point d'exclamation égal** !=.

Collections

Les collections permettent de regrouper des valeurs.

Les collections ont une taille :

```
>>> len("oui")
3
```

str : Chaine de caractères

Une `str` utilise comme séparateur des guillemets simples ' ou doubles ". Il est possible d'échapper un séparateur avec un \, si l'on utilise des ' et des "

```
>>> "J'aime les carottes"
"j'aime les carottes"
>>> >>> 'j\'aime les carottes'
"j'aime les carottes"
```

Quelques opérations sont utilisables avec les `str` :

```
>>> 3 * "po"
'popopo'
>>> "Allo " + 'quoi'
'Allo quoi'
```

list : Liste

Une `list` est ordonnée, accepte le mélange de types et quelques opérateurs. Les éléments d'une liste sont accessibles par leurs positions. Attention, en informatique, on aime bien compter à partir de 0, et non de 1. Dans l'exemple, l'élément 1 désigne le deuxième dans la liste.

```
>>> [1, "hop", 2.5]
[1, 'hop', 2.5]
>>> 3 * [1, 2]
[1, 2, 1, 2, 1, 2]
>>> ["il", "fait"] + ["beau"]
['il', 'fait', 'beau']
>>> ["pomme", "poire", "scoubidou"][1]
'poire'
```

tuple : Tuple

Un `tuple` est une liste immuable (que l'on ne peut pas modifier). Les tuples sont essentiellement utilisés par les fonctions quand elles renvoient plusieurs valeurs.

dict : dictionnaire

Dans une liste, les éléments sont accessibles par leur rang, dans un dictionnaire, par une clef (de n'importe quel type).

```
>>> { "un": 1, "deux": 2}
{'un': 1, 'deux': 2}
>>> { "un": 1, "deux": 2}["deux"]
2
```

ハッキング *Hakkingu*

Les `str` utilisent UTF8, ce qui permet l'utilisation des smileys : "`J'adore` 🐍".

Les listes disposent d'une syntaxe plus souple pour favoriser la lisibilité : il est possible d'utiliser plusieurs lignes etd'avoir une virgule en trop à la fin de la liste (pour faciliter le copier/coller).

```
>>> [1,
... 2,
... ]
[1, 2]
```

Une `str` est une `list` spécialisée dans les lettres.

Des collections de collections sont possibles.

Bien plus de types Python existent [https://docs.python.org/fr/3.13/library/stdtypes.html].

Variables

Une valeur peut être désigné par une variable.

Les variables peuvent stocker le résultat d'une opération ou d'une fonction.

```
>>> a=42
>>> a+3
45
>>> b=1+2
>>> b
3
```

ハッキング Hakkingu

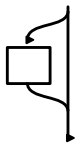
Les variables ont une portée, elles ne sont pas accessibles de partout : une variable crée dans une boucle (un controle de flot), ne sera pas disponible en dehors.

Contrôle de flot

Un programme peut ne pas se contenter d'une exécution séquentielle (une suite d'opérations).

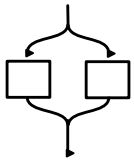
Python a la particularité d'utiliser des indentations pour définir les blocs. Par convention, l'indentation est de quatre espaces. Dans les éditeurs de texte qui reconnaissent Python, une tabulation (la touche "→|" tout à gauche) va créer quatre espaces.

if : Si



```
a = True
if a:
    print("yeaaah")
```

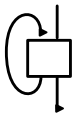
if/else : Si, sinon



```
a = "ok"
if a == "ok":
    print("Oui")
else:
    print("Non")
```

for : Pour

Une boucle for va utiliser un **itérateur**, qui pourra être une **collection**, ou plus simplement la fonction `range`.



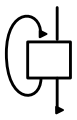
```
>>> for i in range(3):
...     print(i)
...
0
1
2
```

`range(n)` va compter de `0` jusqu'à `n-1`.

```
>>> for lettre in "abc":
...     print(lettre)
...
a
b
c
```

while : Tant que

Parfois, on ne sait combien de fois on va boucler. On va donc itérer jusqu'à ce qu'une clause soit vraie.



```
>>> a = 0
>>> while a < 2:
...     a = a + 1
...
>>> print(a)
2
```

ハッキング *Hakkingu*

En plus de `if` et `else` il y a `elif` contraction de `else if`.

`break` interromps une boucle `for`.

```
>>> for i in range(3000):
...     print(i)
...     if i == 2:
...             break
...
0
1
2
```

`continue` fait passer à l'itération suivante dans une boucle `for`.

Fonctions

Une fonction regroupe un bloc de codes qui pourra être appelée dans une autre partie du code.

Une fonction a une entrée (des `arguments`) et une sortie (le `return`).

Python fournit des fonctions natives [https://docs.python.org/fr/3.13/library/functions.html].

```
>>> # abs: valeur absolue
>>> abs(-2)
2
```

Python est un langage orienté objet, les valeurs sont des objets. Les objets ont des `function` que l'on appelle `method`.

```
>>> # upper: passage en majuscule d'une chaine de caractères.
>>> "bob".upper()
'BOB'
```

Écrire une fonction

Le bloc définissant une fonction est indenté (chaque ligne commence par une tabulation).

```
def hello(name):
    return "Hello {}".format(name)

print(hello("World"))
```

Modules

Les modules permettent de regrouper des fonctions, des variables.

Le nom du module comme préfixe évite les collisions de noms (différents modules peuvent avoir des éléments avec le même nom).

```
>>> import math
>>> math.pi
3.141592653589793
>>> # Python utilise des radians en trigonométrie
>>> math.sin(math.pi / 2)
1.0
```

Des éléments spécifiques d'un module peuvent être importés et utilisés sans préfixe.

```
>>> from math import pi, sin
>>> sin(pi / 2)
1.0
```

Tous les éléments d'un module peuvent être importés d'un coup.

```
>>> from math import *
>>> asin(1) / pi
0.5
```

ハッキング Hakkingu

Il est possible de créer des modules et de les partager.

Bibliothèque standard

Python est "livré avec des piles", il inclut une bibliothèque standard [https://docs.python.org/fr/3.13/library/index.html] conséquente.

Pas d'inquiétudes à voir sur la grande quantité de modules existants, une partie sont ésotériques, et il suffit de chercher par domaine (maths, textes, fichiers, réseaux, etc.) puis d'explorer les quelques modules du domaine.

Bibliothèques tierces

Une multitude de modules sont disponibles sur Internet, un des bienfaits de l'open source.

Le site PyPi [https://pypi.org/] est le portail officiel des modules Python.

Pip

La commande `pip` permet d'installer des modules.

```
pip install requests
```

ハッキング Hakkingu

Les interpréteurs Python en ligne ne disposent pas de tous les modules et les installent de manières différentes.

Utiliser `pip` en dehors d'un environnement virtuel est fortement déconseillé, les souffrances à court terme seront terribles.

Un module peut "tirer" d'autres modules, qui seront aussi installés par `pip`.

Les 600 000 modules disponibles sur PyPi ne sont pas tous de la même qualité.

Environnement virtuel

Chaque projet est différent et nécessite des modules (dans des versions spécifiques) différents.

venv : Virtual Environment

La bibliothèque standard Python dispose du module `venv`.

Création

Dans le terminal, dans le dossier source :

```
python3 -m venv .venv
```

Activation

Même terminal, même dossier.

```
source .venv/bin/activate
```

L'environnement virtuel doit être créé une fois, et activé à chaque session (à chaque fois que l'on ferme la fenêtre du terminal).

ハッキング Hakkingu

Différentes personnes doivent pouvoir travailler sur un même projet, dans un environement virtuel déterministe.

Les éditeurs de code gèrent les environnements virtuels.

Introspection

Python permet de "découvrir" les objets.

help : Aide

```
>>> help(abs)
Help on built-in function abs in module builtins:

abs(x, /)
    Return the absolute value of the argument.
```

dir : Lister les attributs

Inspecter un module permet de découvrir les objets qu'il met à disposition.

```
>>> import math
>>> dir(math)
['__doc__', '__file__', '__loader__', '__name__', '__package__',
'__spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh',
'cbrt', 'ceil', 'comb', 'copysign', 'cos', 'cosh', 'degrees', 'dist',
'e', 'erf', 'erfc', 'exp', 'exp2', 'expm1', 'fabs', 'factorial',
'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf',
'isclose', 'isfinite', 'isinf', 'isnan', 'isqrt', 'lcm', 'ldexp',
'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'nextafter',
'perm', 'pi', 'pow', 'prod', 'radians', 'remainder', 'sin', 'sinh',
'sqrt', 'sumprod', 'tan', 'tanh', 'tau', 'trunc', 'ulp']
>>> help(math.cos)
Help on built-in function cos in module math:

cos(x, /)
    Return the cosine of x (measured in radians).
```

Tout est objet, il est possible de `dir` n'importe quoi.

```
>>> dir(5)
['__abs__', '__add__', '__and__', '__bool__', '__ceil__', '__class__',
'__delattr__', '__dir__', '__divmod__', '__doc__', '__eq__',
'__float__', '__floor__', '__floordiv__', '__format__', '__ge__',
'__getattribute__', '__getnewargs__', '__getstate__', '__gt__',
'__hash__', '__index__', '__init__', '__init_subclass__', '__int__',
```

```
'__invert__', '__le__', '__lshift__', '__lt__', '__mod__', '__mul__',
'__ne__', '__neg__', '__new__', '__or__', '__pos__', '__pow__',
'__radd__', '__rand__', '__rdivmod__', '__reduce__', '__reduce_ex__',
'__repr__', '__rfloordiv__', '__rlshift__', '__rmod__', '__rmul__',
'__ror__', '__round__', '__rpow__', '__rrshift__', '__rshift__',
'__rsub__', '__rtruediv__', '__rxor__', '__setattr__', '__sizeof__',
'__str__', '__sub__', '__subclasshook__', '__truediv__', '__trunc__',
'__xor__', 'as_integer_ratio', 'bit_count', 'bit_length', 'conjugate',
'denominator', 'from_bytes', 'imag', 'is_integer', 'numerator', 'real',
'to_bytes']
```

Les **attributs** avec un nom commençant et finissant par "__" sont dits **magiques**. Dans un premier temps, il faut se concentrer sur les autres.

type : Type d'objet

```
>>> type(5)
<class 'int'>
>>> type(5.0)
<class 'float'>
>>>
```

ハッキング Hakkingu

La fonction `help` n'existe malheureusement pas sur le Python de Numworks.